Node.js Blueprints

Summary

All the examples used in the chapters are basically Node.js applications. So to test them you need Node.js installed. It usually comes with Node's package manager automatically. Here is a list of all the needed software:

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)
- MySQL (v5.5.25)
- MongoDB (v2.4.4)
- PhantomJS (could be downloaded from here http://phantomjs.org)
- SASS (requires Ruby and SASS's gem, here is more info about the installation process http://sass-lang.com/install)

01. Common Programming Paradigms

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)

Images used in the chapter

- picture01.png

Code files provided in the chapter

```
└ code
  └ air.js
  └ car.js
  └ control.js
  └ engine.js
  └ wheels.js
```

Running the example

- navigate to the *code* folder
- run *node car.js*

02. Developing a Basic Site with Node.js and Express

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)
- MySQL (v5.5.25)
- MongoDB (v2.4.4)

Images used in the chapter

- none

Code files provided in the chapter

```
└ code_cli
  └ site
    └ public
      └ stylesheets
        └ style.css
        └ style.less
    └ routes
      └ index.js
    └ views
      └ index.jade
      └ login.jade
└ code_package.json
  └ app.js
  └ package.json
```

Running the examples

The chapter is about Express.js framework. It could be installed by using a command line tool (*code_cli* folder) or via the *package.json* file.

Installing via *package.json*:

- go to *code_package.json* folder
- type *npm install*
- when the installation finishes run *node app.js* in the same folder
- open *http://127.0.0.1:1337/* in a browser to see that the server works

Installing via Express.js's CLI

- install Express.js globally by running

- now you may go to any folder and execute *express --sessions --css less myapp* which will create a boilerplate code for an Express.js application. In *code_cli* folder this is already done. The files there are produced by running the Express.js's CLI.

Running the example produced by the command line tool

- go to *code_cli/site* folder
- run *npm install*
- run *node app.js*
- open *http://127.0.0.1:3000/* in a browser. You should see a login page. Type *admin* for username and *admin* for password. When you log in you should be able to log out.

03. Writing a Blog Application with Node.js and AngularJS

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)
- MySQL (v5.5.25)
- MongoDB (v2.4.4)

Images used in the chapter

- none

Code files provided in the chapter

```
└ code
  └ controllers
    └ api
      └ add.js
      └ delete.js
      └ edit.js
      └ get.js
    └ admin.js
    └ index.js
  └ models
    └ Articles.js
  └ public
    └ styles.css
    └ admin.js
    └ angular.min.js
    └ angular-route.min.js
    └ blog.js
  └ views
    └ admin.jade
    └ layout.jade
    └ list.jade
    └ login.jade
  └ index.js
  └ package.json
  └ nodejsblueprints.sql
└ code_ng_fundamentals
  └ page.html
  └ angular.min.js
  └ angular-route.min.js
  └ HeaderController.js
```

Running the examples

The files in *code_ng_fundamentals* are there just for showing the basis of AngularJS. They don't require anything. Just open *page.html* in a browser. Here are the steps for running the code in *code* folder.

Using MySQL:

- run the MySQL server
- create a database called *nodejsblueprints*
- execute the MySQL command from *nodejsblueprints.sql* file. It will creates the necessary MySQL table
- open */code/models/Articles.js* and make sure that *type = "mysql"* and the MySQL settings are properly set (*mysql_user, mysql_pass, mysql_host, mysql_database*).
- go to the *code* folder
- run *npm install*
- run *node index.js*
- open *http://127.0.0.1:3000/* You should see an empty page
- open *http://127.0.0.1:3000/admin* and type *admin* for username and *pass* for password
- add/edit/remove articles

Using MongoDB

- run the MongoDB server
- open */code/models/Articles.js* and make sure that *type = " mongodb"* and the MongoDB settings are set properly
- go to the *code* folder
- run *npm install*
- run *node index.js*
- open *http://127.0.0.1:3000/* You should see an empty page

- open *http://127.0.0.1:3000/admin* and type *admin* for username and *pass* for password
- add/edit/remove articles

04. Developing a Chat with Socket.IO

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)

Images used in the chapter

- files_structure_7338_04_01.png
- picture_7338_04_02.png

Code files provided in the chapter

```
└ code
   └ css
      └ styles.css
   └ html
      └ page.html
   └ index.js
   └ package.json
```

Running the examples

- navigate to *code* folder
- run *npm install*
- run *node index.js*
- open *http://127.0.0.1:3000/* in two separate tabs (or browsers) and test the chat

05. Creating a To-do Application with BackboneJS

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)

Images used in the chapter

- files_7338_05_01.png

Code files provided in the chapter

```
└ code
   └ css
      └ styles.css
```

```
  └ html
    └ page.html
  └ js
    └ collections
      └ ToDos.js
    └ models
      └ ToDo.js
    └ vendors
      └ backbone.js
      └ jquery-1.10.2.min.js
      └ underscore-min.js
    └ views
      └ add.js
      └ edit.js
      └ list.js
    └ app.js
  └ favicon.ico
  └ index.js
```

Running the examples

- navigate to *code* folder
- run *node index.js*
- open *http://127.0.0.1:3000/*

06. Using Node.js as a Command-line Tool

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)

Images used in the chapter

- 7338_06_01.png
- 7338_06_02.png
- 7338_06_03.png
- 7338_06_04.png
- 7338_06_05.png
- 7338_06_06.png
- 7338_06_07.png

- 7338_06_08.png
- 7338_06_09.png
- 7338_06_10.png

Code files provided in the chapter

```
└ code
  └ images
    └ A
      └ image.png
    └ B
      └ C
        └ image.png
      └ image.jpg
    └ image.png
  └ lib
  └ index.js
  └ package.json
```

Running the examples

- Login into Flickr
- Open *http://www.flickr.com/services/apps/create/apply/*
- Choose *APPLY FOR A NON-COMMERCIAL KEY*
- Create a new App
- open *index.js* and enter your Key and Secret there. Set the value of *oauth_consumer_key* and *oauth_consumer_secret*
- Wait few minutes so Flickr enables your application
- navigate to *code* folder
- run *npm install*
- run *node index.js*
- type *images* as path and press enter
- type *y* and press enter
- a new tab will be opened in your default browser
- click on *OK, I'LL AUTHORIZE IT*
- the browser will redirect to a page where *oauth_token* and *oauth_token_secret* could be seen. They could be used in *index.js* and the authorization will be skipped next time.
- at the same time the console will show you the uploaded files

07. Showing a Social Feed with EmberJS

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)

Images used in the chapter

- 7338_07_01.png
- 7338_07_01_v2.png
- 7338_07_02.png
- 7338_07_03.png
- 7338_07_04.png
- 7338_07_05.png
- 7338_07_06.png
- 7338_07_07.png
- 7338_07_08.png

Code files provided in the chapter

```
└ code
   └ css
      └ styles.css
   └ html
      └ page.html
   └ js
      └ ember-1.3.1.js
      └ handlebars-1.1.2.js
      └ jquery-1.10.2.js
      └ scripts.js
   └ index.js
   └ package.json
```

Running the examples

- go to *code* folder
- run *npm install*
- run *node index.js*
- open *http://127.0.0.1:3000/* and type a valid Twitter handler

08. Developing Web App Workflow with Grunt and Gulp

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)

Images used in the chapter

- 7338_08_01.png
- 7338_08_02.png
- 7338_08_03.png
- 7338_08_04.png
- 7338_08_05.png
- 7338_08_06.png
- 7338_08_07.png
- 7338_08_08.png

Code files provided in the chapter

```
└ code
  └ grunt
    └ build
      └ scripts.js
      └ scripts.min.js
      └ size.log
    └ css
      └ styles.css
    └ custom
      └ generate-manifest.js
      └ jssize.js
    └ docs
      └ contains the generated documentation
    └ img
      └ A.png
      └ B.png
      └ C.png
    └ src
      └ lib
        └ C.js
        └ D.js
      └ A.js
      └ B.js
    └ Gruntfile.js
    └ package.json
    └ cache.manifest
  └ gulp
    └ build
      └ scripts.js
      └ scripts.min.js
      └ size.log
    └ custom
      └ jssize.js
    └ src
      └ lib
        └ C.js
        └ D.js
      └ A.js
      └ B.js
    └ gulpfile.js
    └ package.json
```

Running the examples

Testing Gruntjs

- go to *code/grunt* folder
- run npm install -g grunt-cli
- run *npm install*

- run *grunt*

Testing gulp

- go to *code/gulp*
- run *npm install -g gulp*
- run *npm install*
- run *gulp*

09. Automate Your Testing with Node.js

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)
- PhantomJS (could be downloaded from here http://phantomjs.org)

Images used in the chapter

- from 7338_09_01.png to 7338_09_24.png (24 files)

Code files provided in the chapter

```
└ code
  └ tests
    └ test.spec.js
  └ app.js
  └ file.txt
└ code_dalekjs
  └ tests
    └ dalek.js
  └ screen.jpg
  └ app.js
  └ package.json
└ code_headless
  └ tests
    └ phantom.js
  └ app.js
  └ framework.js
└ code_mocha
  └ tests
    └ test.spec.js
```

└ app.js

└ file.txt

Running the examples

Jasmine

- go to *code* folder
- run *npm install -g jasmine-node*
- run *jasmine-node ./tests*

Dalekjs

- go to *code_dalekjs* folder
- run *npm install -g dalek-cli*
- run *npm install*
- open a new terminal on the same place and run *node app.js*. This will run a server which we are going to test
- in the old terminal run *dalek .\tests\dalek.js -b chrome*

Mocha

- go to *code_mocha* folder
- run *npm install -g mocha*
- run *mocha ./tests*

PhantomJS

- install PhantomJS
- go to *code_headless*
- open another terminal and go to the same folder
- run *node app.js*.
- in the first console run *phantomjs .\tests\phantom.js*

10. Writing Flexible and Modular CSS

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)

Images used in the chapter

- 7338OS_10_01.png
- 7338OS_10_02.png
- 7338OS_10_03.png

Code files provided in the chapter

└ code

└ absurd

└ login.css

```
    └ styles.css
    └ incode.js
    └ styles.js
  └ absurd-loginform
    └ src
      └ login.js
    └ login.html
    └ login.css
    └ compile.sh
  └ less
    └ styles.less
  └ sass
    └ styles.scss
  └ stylus
    └ styles.css
    └ styles.styl
```

Running the examples

Absurd

- go to *code/absurd* folder
- run *npm install -g absurd*
- run *npm install absurd*
- run *node ./incode.js* and you should see CSS styles in the console
- run *absurd -s ./styles.js* and you should see CSS styles in the console

Absurd (login form)

- go to *code/absurd-loginform*
- run *absurd -s ./src/login.js -o ./login.css -w ./src/login.js* Absurd runs in a watching mode and every change in *login.js* should update *login.css*.

LESS

- go to *code/less* folder
- run *npm install -g less*
- run *lessc ./styles.less* and you should see the compiled classes

SASS

- you should have SASS installed (check out http://sass-lang.com/install)
- go to *code/sass* folder
- run *sass ./styles.scss* and you should see the compiled CSS styles in the console

Stylus

- go to *code/stylus* folder
- run *npm install -g stylus*
- run *stylus ./styles.styl* and the preprocessor will produce *styles.css* file in the same directory

11. Writing a REST API

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)
- MongoDB (v2.4.4)

Images used in the chapter

- 7338_11_01.png
- 7338_11_02.png
- 7338_11_03.png

Code files provided in the chapter

```
└ code
  └ index.js
  └ responder.js
  └ router.js
  └ test.spec.js
  └ package.json
```

Running the examples

- run the MongoDB server
- go to *code* folder
- run *npm install*
- run *node index.js*
- open another terminal in the same folder and execute *jasmine ./test.spec.js*. You should see all the test passing

12. Developing Desktop Apps with Node.js

Needed software

- Node.js (v0.10.26)
- Node Package Manager / npm (v1.4.3)
- node-webkit (v2.4.4)

Images used in the chapter

- 7338_12_01.png
- 7338_12_02.png
- 7338_12_03.png
- 7338_12_04.png
- 7338_12_05.png
- 7338_12_06.png
- 7338_12_07.png
- 7338_12_08.png
- 7338_12_09.png

Code files provided in the chapter

```
└ code
  └ app
    └ css
      └ the css styles of the application
    └ empty
      └ A
      └ B
    └ js
      └ imageviewer.js
      └ scripts.js
    └ node_modules
    └ image.html
    └ index.html
    └ package.json
```

Running the examples

- download the node-webkite executable (nw) from here https://github.com/rogerwang/node-webkit
- go to *code/app* folder
- run *nw ./*